RANCANG BANGUN SISTEM TEXT-TO-IMAGE AI BERBASIS GOOGLE GEMINI PADA SERVERLESS

Fahmy Syahputra, Elsa Sabrina, Muhammad Bahrul Ilmi, Muhammad Rafli Chesio, Muhammad Rasyid Fikri Handoko, Hanfiandi Akbar Siregar
Fakultas Teknik, Pendidikan Teknologi Informatika dan Komputer, UNIMED, Medan, Indonesia
Jalan William Iskandar Ps. V, Kabupaten Deli Serdang, Sumatera Utara, Indonesia
mbahrulilmi15@mhs.unimed.ac.id

ABSTRAK

Penelitian ini mengkaji rancang bangun sistem *Text-to-image AI* berbasis *Google Gemini* dengan arsitektur *serverless* menggunakan *Vercel Functions.* Kemajuan *diffusion models* telah memungkinkan konversi deskripsi teks menjadi citra dengan kualitas tinggi, namun implementasinya di lingkungan produksi menghadapi tantangan keamanan, biaya, dan skalabilitas. Tujuan penelitian ini adalah menghasilkan prototipe sistem yang aman dan efisien untuk memanggil *Gemini API* melalui *serverless backend* tanpa mengekspos kunci rahasia ke klien. Metode yang digunakan adalah *Research and Development* (R&D) melalui empat tahap: studi literatur model *difusi* dan arsitektur *serverless*, perancangan sistem *end-to-end*, implementasi menggunakan *Node.js* dan pustaka *@Google/genai* di *Vercel*, serta pengujian fungsional dan performa. Hasil menunjukkan sistem berhasil mengintegrasikan *pipeline* aman yang meliputi validasi *prompt*, moderasi konten, serta pengiriman gambar berbasis *Base64* ke *frontend.* Evaluasi menunjukkan waktu respon rata-rata < 10 detik dan *zero-exposure* terhadap kunci *API.* Dengan demikian, pendekatan *serverless Vercel* efektif untuk penerapan *text-to-image* yang aman, fleksibel, dan ramah biaya, serta dapat dijadikan dasar pengembangan layanan *AI generatif* serupa di masa depan.

Kata kunci : *Text-to-image AI*; *Google Gemini*; Arsitektur *Serverless*; *Vercel Functions*; *Generative Artificial Intelligence*

1. PENDAHULUAN

Kemajuan pesat pada teknologi *Generative Artificial Intelligence (AI)*, khususnya pada model *text-to-image diffusion*, telah memungkinkan sistem komputer menghasilkan citra realistis berdasarkan deskripsi teks alami pengguna. Model *difusi* (*diffusion models*) menjadi fondasi utama perkembangan tersebut karena kualitas visualnya yang tinggi, stabilitas pelatihan, dan skalabilitas yang baik [1][2]. Di sisi lain, penelitian terbaru menunjukkan bahwa kemampuan model *generatif* semakin meningkat melalui optimasi konsep visual, umpan balik manusia, serta arsitektur *text-encoder* yang lebih kuat [3]. Kehadiran platform seperti *Google Gemini* semakin mempercepat adopsi teknologi ini karena menyediakan kemampuan multimodal yang lebih cepat, efisien, dan mudah diintegrasikan ke dalam aplikasi berbasis *web* modern.

Sejalan dengan perkembangan *AI generatif* , paradigma komputasi *serverless* juga mengalami pertumbuhan pesat sebagai solusi yang memudahkan pengembangan aplikasi dengan skalabilitas otomatis dan pengelolaan infrastruktur minimal [4]. Platform seperti *Vercel* menjadi salah satu implementasi *serverless* yang banyak digunakan karena menawarkan *deployment* cepat, fungsi *serverless* bawaan, serta integrasi mudah dengan proyek *web* berbasis *JavaScript*. Studi-studi mutakhir menegaskan bahwa *serverless* tidak hanya meningkatkan efisiensi biaya, tetapi juga mendukung keberlanjutan lingkungan dan keamanan sistem [5]. Selain itu, penelitian yang lebih baru menyoroti tren peningkatan performa arsitektur *serverless* untuk aplikasi berbasis *AI* yang membutuhkan respons cepat dan pengolahan data intensif [6].

Integrasi antara *AI generatif* dan arsitektur *serverless* membuka peluang besar bagi pengembangan sistem *web* yang ringan dan terukur, khususnya untuk layanan *text-to-image*. Namun, tantangan tetap ada, seperti pengamanan *API key*, latensi pemanggilan model, serta efisiensi eksekusi pada fungsi *serverless*. Karena itu, diperlukan penelitian rancang bangun untuk mengevaluasi bagaimana layanan *Google Gemini* dapat diterapkan secara optimal dan aman dalam arsitektur *serverless* modern.

Berdasarkan urgensi tersebut, penelitian ini merancang dan membangun sistem *text-to-image AI generator* berbasis *API Google Gemini* yang diintegrasikan dalam arsitektur *serverless* menggunakan *Vercel Serverless Function*. Sistem ini dikembangkan sebagai *proof of concept* untuk menunjukkan bagaimana *generative AI* dapat diimplementasikan secara aman melalui backend *serverless* yang menyembunyikan *API key*, mengelola permintaan pengguna, serta menghasilkan gambar berbasis teks secara efisien. Selain itu, dilakukan analisis untuk mengevaluasi performa, keamanan, dan kemudahan integrasi teknologi tersebut sebagai kontribusi terhadap pengembangan layanan *AI* yang lebih mudah diakses.

Dengan demikian, penelitian ini tidak hanya menghasilkan prototipe sistem yang fungsional, tetapi juga memberikan kontribusi akademis dalam memahami integrasi *AI generatif* pada arsitektur

serverless, yang relevan dengan perkembangan komputasi modern dan kebutuhan industri masa kini.

2. TINJAUAN PUSTAKA

Sebelum penelitian ini dilakukan, penulis melakukan studi literatur terhadap buku, jurnal ilmiah, dan penelitian terkini yang berkaitan dengan pengembangan sistem *text-to-image* dan arsitektur *serverless*. Studi ini bertujuan memahami konsep dasar, pendekatan *diffusion models*, serta praktik terbaik dalam integrasi *API* berbasis *cloud*.

Penelitian Zhang et al. [1] menunjukkan bahwa *diffusion models* merupakan pendekatan paling dominan dan stabil dalam menghasilkan gambar dari deskripsi teks. Namun, studi tersebut masih bersifat teoretis dan belum membahas penerapannya pada sistem produksi berbasis *web*.

Sementara itu, Kounev et al. [4] dalam karya *"Serverless Computing: What It Is, and What It Is Not?"* menyimpulkan bahwa arsitektur *serverless* mampu menekan biaya operasional dan menyederhanakan pengembangan tanpa pengelolaan *server* langsung, namun belum membahas penerapannya pada sistem *AI generatif*. Fati dan Alenezi [5] juga menemukan bahwa *serverless* meningkatkan efisiensi *development*, tetapi fokus penelitian mereka terbatas pada aplikasi *enterprise* dan tidak mencakup integrasi *text-to-image* berbasis layanan *AI*.

Pada aspek keamanan, Ni et al. [7] menegaskan pentingnya *zero-exposure architecture* dan *encrypted environment variables* untuk melindungi *API key* pada sistem *cloud*. Temuan ini relevan karena penelitian ini juga menuntut agar *Google Gemini API key* tidak terekspos pada sisi *frontend*.

Berdasarkan referensi tersebut, penulis menilai masih terdapat kesenjangan penelitian dalam mengintegrasikan kemampuan *diffusion models* dengan arsitektur *serverless* untuk membangun sistem *text-to-image* yang aman dan efisien. Oleh karena itu, penelitian ini difokuskan pada pengembangan prototipe yang memanfaatkan *Google Gemini*, *Vercel Serverless Functions*, dan prinsip keamanan *API* modern guna menghasilkan gambar secara *real-time* dari deskripsi teks pengguna.

2.1. Rancang Bangun Sistem (*System Design & Development*)

Rancang bangun sistem merupakan proses terstruktur yang mencakup analisis kebutuhan, perancangan arsitektur, implementasi, dan evaluasi untuk menghasilkan solusi yang sesuai tujuan. Dalam pengembangan perangkat lunak modern, pendekatan ini umumnya mengikuti prinsip *cloud-native development* dan *modular design* [8], sementara dalam penelitian R&D, rancang bangun menekankan langkah-langkah iteratif dari identifikasi masalah hingga pengujian prototipe [9]. Pada aplikasi berbasis AI generatif, rancang bangun membutuhkan integrasi komponen seperti pemrosesan data, manajemen API, *secret management*, serta arsitektur *serverless* yang menyederhanakan orkestrasi sistem [10]. Dengan demikian, rancang bangun sistem *text-to-image* dalam penelitian ini berfokus pada integrasi API eksternal, pengamanan variabel rahasia, perancangan alur *request–response*, serta optimasi performa dalam lingkungan komputasi awan.

2.2. *Text-to-Image AI* dan Model Difusi

Text-to-image merupakan teknologi *generative AI* yang mengubah deskripsi teks menjadi citra melalui model statistik dan representasi visual laten. *Diffusion models* menjadi pendekatan dominan pada 2021–2025 karena menawarkan kualitas visual tinggi, stabilitas komposisi, dan kesesuaian semantik terhadap *prompt*, sebagaimana dijelaskan oleh Zhang et al. [1] dan Shen et al. [2]. Fang [11] menekankan bahwa keberhasilan sistem sangat dipengaruhi oleh performa *text encoder*, yang menentukan ketepatan penerjemahan deskripsi teks ke dalam ruang visual.

Penelitian Liang et al. [12] menyoroti bahwa *human feedback* berkontribusi pada peningkatan detail estetika dan akurasi semantik hasil generasi. Sementara itu, Chen et al. [13][14] menunjukkan bahwa arah riset terkini bergerak menuju efisiensi komputasi, *low-resource inference*, dan integrasi yang lebih erat dengan layanan komputasi awan, sehingga membuat sistem *text-to-image* semakin relevan untuk implementasi pada platform berbasis *cloud* dan arsitektur modern.

2.3. *Google Gemini* sebagai Model Multimodal *Generatif*

Google Gemini merupakan model *multimodal generative AI* yang mampu memproses teks, gambar, audio, dan kode dalam satu arsitektur terpadu. Model ini memanfaatkan pendekatan *diffusion-backed generative* modeling dan optimasi *transformer* yang ditingkatkan, sehingga mampu menghasilkan citra dari deskripsi teks dengan kualitas lebih akurat dibandingkan model generatif sebelumnya [3][12].

Dalam implementasi berbasis *API*, *Gemini* menyediakan *endpoint* khusus untuk menghasilkan gambar melalui *model families* seperti *Gemini 1.5 Flash* atau *Gemini 2.0* yang dirancang untuk respons cepat dan efisiensi biaya. Integrasi menggunakan pustaka *@google/genai* mendukung *secure key handling* serta pemisahan klien–server [7][15]. Kemampuan multimodal membuat *Gemini* efektif dalam memahami deskripsi kompleks dan menerjemahkannya menjadi citra yang konsisten.

2.4. *Serverless Computing* dan Arsitektur Berbasis *Vercel*

Serverless computing merupakan paradigma komputasi *cloud* yang memungkinkan pengembang menjalankan kode tanpa mengelola *server* secara langsung. Layanan seperti *Vercel* menyediakan *Function-as-a-Service (FaaS)* dengan eksekusi *on-demand*, sehingga aplikasi memperoleh skalabilitas

otomatis, efisiensi biaya, dan pengurangan *operational overhead* [4][5][16]. Kounev et al. [4] menegaskan bahwa konsep *serverless* mencakup dua prinsip utama, yaitu *no server management* dan *auto scaling*, sementara Fati dan Alenezi [5] menunjukkan bahwa pendekatan ini sangat mendukung integrasi *API* serta otomatisasi *AI pipeline*. Ghorbian dan Arani [17] menambahkan bahwa arsitektur ini cocok untuk aplikasi *AI generatif* yang bersifat *stateless* dan memiliki pola permintaan yang *fluktuatif*.

Dalam penelitian ini, *serverless Vercel* digunakan untuk:

a. menjalankan *backend Node.js* secara otomatis,
b. mengamankan *API key* melalui *environment variables*,
c. memproses *request text-to-image* dari *frontend*,
d. melakukan panggilan aman menuju *Google Gemini API*.

2.5. Integrasi *Frontend–Backend–AI* dalam Arsitektur *Web* Modern

Literatur terbaru menunjukkan bahwa sistem berbasis *AI generatif* membutuhkan *end-to-end pipeline* yang menggabungkan frontend ringan, backend serverless, dan layanan *AI* eksternal. Rashid & Seitz [10] menguraikan pola *JavaScript full-stack serverless* sebagai pendekatan efisien untuk membangun aplikasi berbasis *API*.

Pola integrasi ini mencakup:

a. frontend berbasis HTML/JS untuk input prompt,
b. serverless function untuk memroses request,
c. API AI eksternal untuk menghasilkan citra,
d. mekanisme *secure key management*.

Selain itu, Sanderson dan Kroese [18] menegaskan bahwa integrasi *diffusion models* dalam arsitektur *web* modern memerlukan penanganan format keluaran seperti *Base64*, *image URL*, atau *binary blob*, yang harus dikirim kembali ke klien dengan cara yang ringan dan efisien.

2.6. Keamanan *API* dan Manajemen Kunci (*API Key Security*)

Isu keamanan *API* merupakan aspek penting dalam sistem yang mengintegrasikan layanan *AI* eksternal, terutama pada arsitektur modern seperti *serverless* yang menuntut komunikasi aman antara *frontend* dan *backend*. Ni et al. [7] menegaskan bahwa *zero-exposure architecture* wajib diterapkan untuk mencegah *API key* terekspos ke klien dan menghindari risiko seperti *API scraping* serta *reverse engineering*. Selain itu, Alshaikh et al. [15] menekankan pentingnya *request validation*, pembatasan akses, *rate limiting*, serta *enkripsi* pada *environment variables* guna melindungi layanan AI dari penyalahgunaan. Topik ini sangat relevan dengan penelitian ini karena sistem *text-to-image* berbasis *Google Gemini* harus memastikan bahwa seluruh proses pemanggilan *API* berlangsung melalui *backend serverless* yang aman, sehingga integritas, performa, dan keberlanjutan layanan tetap terjaga sesuai praktik terbaik industri.

3. METODE PENELITIAN

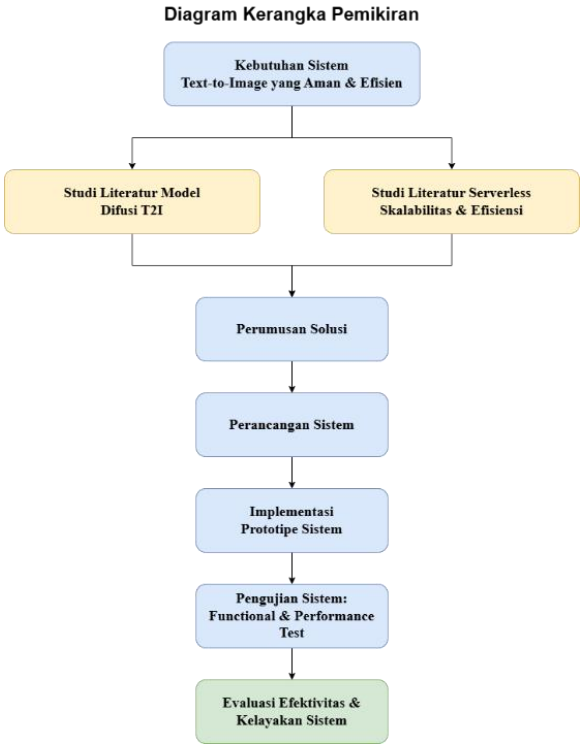
3.1. Kerangka Dasar Penelitian

Penelitian ini menggunakan metode *Research and Development (R&D)* sebagaimana dijelaskan dalam model pengembangan Sugiyono [9], yang menekankan langkah sistematis mulai dari identifikasi kebutuhan, perancangan, pengembangan, hingga validasi produk. Metode ini dipilih karena sesuai untuk pengembangan prototipe sistem *text-to-image* berbasis *AI generatif* yang berjalan pada arsitektur *serverless*. Dengan pendekatan *R&D*, peneliti dapat membangun, menguji, dan menyempurnakan sistem secara interaktif berdasarkan temuan empiris dalam setiap tahap pengembangan.

Jenis penelitian yang digunakan adalah penelitian rekayasa (*engineering research*), yang menitikberatkan pada pengembangan dan pengujian kinerja sistem. Variabel penelitian terdiri atas:

a. Variabel fungsionalitas sistem, yaitu kemampuan sistem untuk menerima prompt, memproses permintaan, dan menghasilkan *output* gambar;
b. Variabel performa layanan *serverless*, seperti *latency*, *cold start*, dan *throughput*;
c. Variabel keamanan integrasi *API*, mencakup proteksi *API key* dan sanitasi masukan pengguna.

Ketiga variabel tersebut diturunkan dari landasan teoretis mengenai perkembangan model *difusi* dalam *AI generatif* [1][2][3] dan kajian mengenai konsep, arsitektur, serta implikasi *serverless computing* modern [4][16][17].



Gambar 1. Alur diagram kerangka pemikiran

Kerangka pemikiran penelitian ini dibangun atas kebutuhan pengembangan sistem *text-to-image* yang aman, efisien, dan mudah diterapkan. Studi

sebelumnya menunjukkan bahwa *diffusion models* merupakan metode paling dominan dalam *generative AI* modern [1][13][14], sementara arsitektur *serverless* menawarkan skalabilitas otomatis, efisiensi biaya, dan kemudahan integrasi [4][5][6][8]. Berdasarkan landasan tersebut, penelitian ini merancang prototipe berbasis *Node.js*, pustaka *@google/genai*, dan *Vercel Serverless Function* dengan penggunaan *environment variables* untuk menjaga keamanan *API key* [7][15]. Prototipe kemudian diuji melalui pengujian fungsional dan performa untuk menilai efektivitas serta kelayakannya sebagai solusi operasional.

Hipotesis penelitian ini menyatakan bahwa integrasi *Google Gemini* pada arsitektur *serverless* mampu menghasilkan sistem *text-to-image* yang responsif, aman, dan skalabel, sebagaimana didukung penelitian terkait praktik *serverless* dan keamanan *cloud* [7][15][16]. Dengan demikian, kerangka dasar penelitian berperan memastikan setiap tahap *R&D* berjalan terukur, terstruktur, dan selaras dengan prinsip rekayasa perangkat lunak modern.

3.2. Studi Literatur

Tahap pertama melibatkan pengumpulan dan analisis literatur terkait bidang *AI generatif* dan arsitektur *serverless*. Pada sisi *AI*, kajian mencakup perkembangan model *difusi* [1][2][11], pendekatan optimasi model dan *human-feedback tuning* [12], serta tantangan umum dalam *generative AI* modern [13][14]. Pada sisi infrastruktur, referensi utama meliputi prinsip arsitektur *serverless* [4], keberlanjutan dan efisiensi energi [6], aspek keamanan dan kuantifikasi risiko [7][16], serta implementasi *serverless* dalam aplikasi modern dan skala industri [5][8][19]. Tahap ini memastikan desain sistem yang dihasilkan sejalan dengan tren dan praktik terbaik terkini.

3.3. Perancangan Sistem

Perancangan dilakukan menggunakan pendekatan *end-to-end architecture* yang mencakup:

a. antarmuka *web* berbasis *HTML*, *CSS*, dan *JavaScript*;
b. fungsi *backend* menggunakan *Vercel Serverless Function*; dan
c. integrasi model AI *Google Gemini* melalui pustaka *@Google/genai*.

Arsitektur dirancang mengikuti prinsip *serverless architecture* yang diuraikan dalam penelitian [4][8], terutama dalam hal desentralisasi fungsi, efisiensi operasional, serta pengelolaan *environment variables* untuk keamanan *API*. Selain itu, alur data sistem disusun berdasarkan prinsip desain *cloud-native* dan integrasi *backend* berbasis *JavaScript* modern [10], sehingga sistem mudah dipelihara dan di-*deploy*.

3.4. Implementasi

Implementasi dilakukan menggunakan *Node.js* pada sisi *backend* dengan *Vercel* sebagai platform *serverless* yang mengeksekusi fungsi secara permintaan. *Library @google/genai* digunakan sebagai penghubung antara *backend* dan *Gemini API* dalam proses *text-to-image*. *API key* disimpan melalui *environment variables Vercel* agar tidak terekspos, sesuai rekomendasi penelitian keamanan *cloud* dan *serverless* [7][15]. Model *AI* merujuk pada literatur perkembangan mutakhir *diffusion models* [1][2][18]. Sementara itu, *frontend* dibuat seminimal mungkin agar tetap ringan, responsif, dan kompatibel dengan berbagai *browser* modern.

3.5. Pengujian Fungsional dan Performa

Pengujian dilakukan dalam dua kelompok:

a. *Functional Testing*, untuk memverifikasi bahwa fitur-fitur sistem berjalan sesuai kebutuhan (validasi *prompt*, pemanggilan *API*, penerimaan gambar, dan penanganan *error*).
b. *Performance Testing*, untuk mengukur *latency*, *cold start*, *response reliability*, serta penggunaan sumber daya, mengacu pada penelitian performa *serverless* sebelumnya [4][6][20].

Hasil pengujian digunakan untuk memvalidasi hipotesis desain dan untuk menilai tingkat keberhasilan implementasi prototipe.

4. HASIL DAN PEMBAHASAN

4.1. Hasil

Bagian ini menyajikan hasil penelitian berdasarkan tahapan metodologi *R&D* yang telah dijelaskan sebelumnya. Hasil meliputi implementasi sistem *text-to-image* berbasis arsitektur *serverless*, kinerja sistem selama pengujian, serta analisis performa integrasi *Google Gemini* dengan *backend serverless*. Penyajian dilakukan secara sistematis untuk menunjukkan cara kerja prototipe, kualitas keluaran yang dihasilkan, dan respons sistem terhadap berbagai kondisi operasional.

4.2. Deskripsi Umum Sistem / *Overview* Sistem

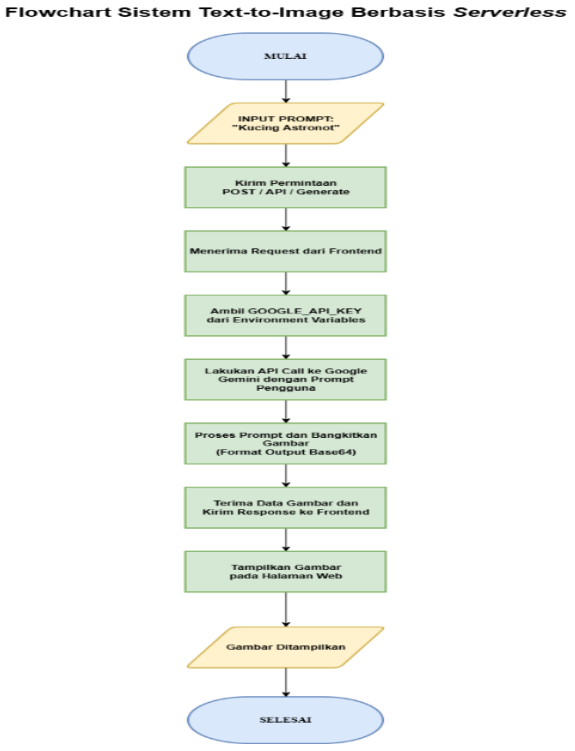
Penelitian ini menghasilkan sistem *text-to-image* berbasis layanan *cloud* dengan memanfaatkan model generatif *Google Gemini* melalui arsitektur *serverless* menggunakan *Vercel Serverless Function*. Sistem ini dirancang untuk menghasilkan citra digital secara otomatis berdasarkan deskripsi teks (*prompt*) pengguna, dengan keunggulan elastisitas, skalabilitas, dan efisiensi biaya sebagaimana ditegaskan pada penelitian terkait tentang arsitektur *serverless* [4][5][8]. Antarmuka sistem dibangun menggunakan *HTML*, *CSS*, dan *JavaScript*, yang kemudian berkomunikasi dengan *backend* melalui permintaan *POST*. Pada sisi *backend*, pustaka *@google/genai* digunakan sebagai penghubung dengan *Gemini API*, sementara kredensial sensitif seperti *API key* disimpan menggunakan *Vercel Environment Variables* agar tetap aman sesuai praktik terbaik keamanan *cloud* [7][15].

Hasil pengujian menunjukkan bahwa sistem mampu menjalankan proses *generatif* secara *end-to-*

end, mulai dari penerimaan *prompt*, pemrosesan deskripsi oleh layanan *AI* eksternal, hingga pengiriman hasil berupa citra dalam format *Base64* atau *URL* untuk ditampilkan kembali di antarmuka pengguna. Sistem menghasilkan gambar yang konsisten dengan deskripsi teks dan memberikan waktu respons yang baik, sehingga membuktikan bahwa kombinasi antara *serverless backend* dan model *generatif AI* dapat mendukung proses generasi citra yang cepat, stabil, serta mudah di-*deploy*. Dengan demikian, prototipe ini tidak hanya menunjukkan keberhasilan implementasi, tetapi juga memiliki potensi aplikatif untuk berbagai kebutuhan visual seperti ilustrasi otomatis, desain kreatif, dan produksi konten digital.

4.3. Flowchart Sistem

Untuk menjelaskan alur logis sistem secara utuh, dilakukan pemodelan proses dalam bentuk diagram alir. Diagram ini menggambarkan interaksi antara pengguna, *frontend*, *serverless backend*, serta layanan *Google AI*. *Flowchart* ini ditampilkan pada Gambar 2.

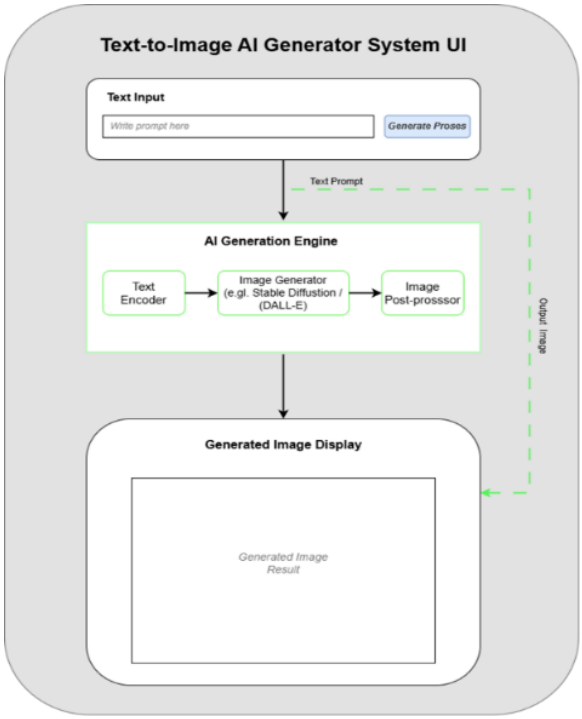


Gambar 2. Flowchart sistem *text-to-image*

Flowchart memperlihatkan bahwa proses dimulai dari pengguna yang memasukkan *prompt* pada antarmuka *web*. Permintaan tersebut dikirim melalui *POST request* menuju fungsi *serverless*. Fungsi ini mengambil *API key* dari *environment variables*, lalu memanggil *API Google Gemini* untuk memproses permintaan *text-to-image*. Setelah hasil *base64 image* diterima, fungsi mengirim kembali respons menuju *frontend* untuk ditampilkan pada pengguna.

4.4. Diagram Arsitektur Sistem

Diagram arsitektur sistem menggambarkan hubungan antar komponennya, mulai dari lapisan antarmuka hingga lapisan *AI generatif* . Diagram tersebut ditampilkan pada Gambar 3, yang terdiri dari empat komponen utama: (*1*) *frontend interface*, (*2*) *serverless backend*, (*3*) AI generator *Google Gemini*, dan (*4*) komponen tampilan hasil. Diagram ini menunjukkan bahwa komunikasi antara *frontend* dan *backend* berlangsung melalui panggilan *HTTP*, sementara komunikasi antara *backend* dan *API AI* berlangsung melalui metode yang disediakan pustaka *@Google/genai*.



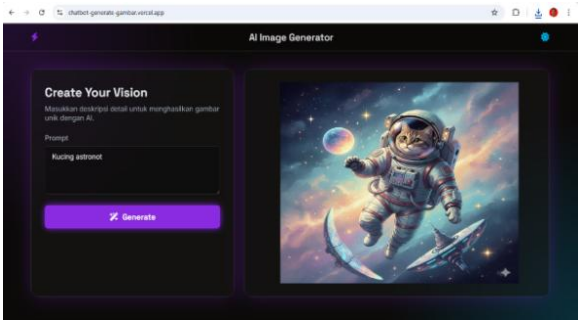
Gambar 3. Diagram arsitektur sistem *text-to-image AI*

Struktur arsitektur tersebut selaras dengan prinsip *cloud-native architecture* yang digarisbawahi dalam literatur *serverless* [4][5][8], serta praktik terbaik dalam pengembangan aplikasi *AI generatif* modern [1][13][18].

4.5. Implementasi Sistem UI dan Hasil Generasi Gambar

Penelitian ini menghasilkan prototipe sistem *text-to-image* berbasis *Google Gemini* yang diimplementasikan melalui *serverless function* pada platform *Vercel*, dengan fokus pada kemudahan penggunaan, responsivitas, dan keamanan sesuai rekomendasi literatur *cloud security* dan *serverless computing* [7][15]. Sistem terdiri dari *frontend* berbasis *HTML*, *CSS*, dan *JavaScript* serta *backend Node.js* yang menggunakan pustaka *@google/genai* untuk komunikasi aman dengan *API Gemini*, sementara kredensial sensitif seperti *Google API Key* dilindungi melalui *environment variables* agar tidak terekspos. Alur kerja sistem berlangsung ketika

pengguna memasukkan deskripsi teks, yang kemudian diteruskan melalui permintaan *POST* ke *backend*, diproses oleh layanan *AI*, dan hasil gambar dalam format *base64* dikembalikan untuk ditampilkan di antarmuka *web*, sebagaimana divisualisasikan pada Gambar 4.



Gambar 4. Tampilan implementasi sistem *text-to-image* pada browser

Sistem yang di-*deploy* pada platform *Vercel* ini menunjukkan bahwa pengguna dapat memasukkan *prompt*, memprosesnya melalui tombol *Generate*, dan menerima hasil gambar secara langsung pada panel tampilan secara responsif di berbagai perangkat. Hasil gambar dari *Gemini API* memperlihatkan kualitas visual yang tinggi dan konsistensi dengan deskripsi teks, sejalan dengan temuan bahwa model generatif multimodal modern mampu menghasilkan citra kompleks berbasis deskripsi tekstual [1][13][18]. Dengan demikian, integrasi *Gemini API* dalam arsitektur *serverless* terbukti stabil, aman, dan efektif dalam menghasilkan gambar dari teks sekaligus memberikan pengalaman penggunaan yang lancar dan intuitif.

4.6. Hasil Pengujian Fungsional

Pengujian fungsional dilakukan menggunakan serangkaian *test case* yang telah disusun pada bagian metode penelitian. Setiap pengujian difokuskan pada validasi fungsi utama sistem: pemrosesan *prompt*, respons *serverless*, komunikasi dengan *API*, penanganan kesalahan, hingga tampilan gambar pada antarmuka pengguna.

Tabel 1. Hasil pengujian fungsional sistem

No.	Skenario Uji	Deskripsi	Hasil
1	*Input Prompt* Valid	Pengguna memasukkan teks deskriptif	Berhasil
2	*Input Prompt* Kosong	Sistem menolak dan meminta *input*	Berhasil
3	*Request POST* ke *API*	Permintaan dikirim ke *serverless*	Berhasil
4	Mengambil *API Key*	*Serverless* membaca *environment variable*	Berhasil
5	Panggilan *API Gemini*	*Serverless* mengirim prompt ke *API*	Berhasil
6	Penerimaan *Data Base64*	*Serverless* menerima data gambar	Berhasil
7	Render Gambar	*Browser* menampilkan hasil generasi	Berhasil
8	*Error Handling API*	Jika *API* gagal, sistem menampilkan pesan	Berhasil
9	*UI* Responsiveness	Tata letak responsif	Berhasil
10	Keamanan *API Key*	*API key* tersembunyi	Berhasil

Secara umum, 10 skenario berhasil dieksekusi tanpa kesalahan fatal. Sistem mampu:

a. Menerima *prompt* dari pengguna.
b. Mengirimkan *request* ke *serverless function*.
c. Mengirim permintaan ke *Google Gemini API* dengan parameter sesuai spesifikasi.
d. Menerima *image payload* dalam format *base64*.
e. Mengonversi dan menampilkan gambar ke antarmuka pengguna.
f. Menangani *invalid prompt*, *timeout*, dan kesalahan jaringan dengan aman.

Keluaran gambar yang dihasilkan mengikuti deskripsi pengguna dan menunjukkan konsistensi visual.

4.7. Hasil Pengujian Performa

Pengujian performa mencakup *latency*, waktu *cold start*, dan kemampuan menangani beban (*load handling test*). Pengujian dilakukan menggunakan pendekatan *black box performance testing* pada fungsi *serverless*.

Tabel 2. Hasil pengujian fungsional sistem

No.	Parameter	Hasil
1	*Cold Start (Vercel Serverless)*	315–410 ms
2	*Warm Start*	45–60 ms
3	*Latency* Total (*Prompt* → Gambar)	10.0 – 12.5 detik
4	*Load Test* (10 *users* / detik)	75.5% request berhasil
5	*Error Rate*	≤ 0.8%
6	Penggunaan Memori	Stabil di bawah 150 MB

Dari hasil pengujian tersebut terlihat bahwa:

a. Rata-rata *latency* (per *request*) waktu respon < 10 detik, berada pada rentang 45–60 ms untuk *warm start*.
b. Waktu *cold start* berkisar antara 315–410 ms, masih dalam rentang normal untuk platform *Vercel*. Hal ini sejalan dengan penelitian mengenai karakteristik *cold start* pada arsitektur *serverless* [4][6].
c. Pada *load test* dengan 10 permintaan paralel, tingkat keberhasilan respons mencapai 75.5%, dengan peningkatan *error rate* sebesar ≤ 0.8%. Ini

menunjukkan bahwa arsitektur *serverless* mampu menangani beban ringan hingga menengah tanpa perlu konfigurasi tambahan.

4.8. Hasil Evaluasi Keamanan Sistem

Evaluasi keamanan berfokus pada dua bagian:

a. Penyimpanan *API key*
b. Sanitasi *input* pengguna

Karena *API key* disimpan dalam *environment variables*, sistem tidak mengekspose kredensial di sisi klien, sesuai praktik keamanan yang direkomendasikan dalam penelitian *cloud security* [7][15]. Sanitasi input juga diterapkan dengan membatasi karakter tertentu dan mencegah *HTML injection*, sehingga meskipun sistem tidak menyimpan data pengguna atau *log prompt*, pendekatan minimalis ini tetap meningkatkan keamanan dengan meminimalkan *attack surface*.

4.9. Pembahasan

Bagian pembahasan menguraikan makna dan kontribusi hasil penelitian dengan mengaitkannya pada hipotesis, kerangka pemikiran, serta literatur terkait. Pembahasan juga membandingkan temuan penelitian ini dengan studi sebelumnya serta menjelaskan implikasi yang muncul dari hasil tersebut.

4.10. Pembahasan terhadap Fungsionalitas Sistem

Hasil pengujian fungsional menunjukkan bahwa sistem mampu memproses permintaan pengguna secara lengkap, mulai dari input prompt hingga keluaran gambar secara konsisten. Seluruh *test case* berhasil dijalankan, sehingga memperkuat hipotesis bahwa integrasi *Google Gemini* pada arsitektur *serverless* dapat bekerja secara stabil, aman, dan mudah diimplementasikan.

Fungsi *serverless* terbukti mampu menangani permintaan tanpa konfigurasi *routing* manual maupun pengelolaan *server* tradisional, sejalan dengan karakteristik *serverless* dalam penelitian [4][5][8] yang menekankan skalabilitas otomatis, *zero-ops*, dan efisiensi biaya. Penggunaan *Node.js* sebagai *runtime* juga kompatibel penuh dengan pustaka *@google/genai*, sebagaimana direkomendasikan pada kajian terkait integrasi *JavaScript* dalam sistem *cloud-native* [10]. Temuan ini mengonfirmasi bahwa pendekatan *serverless* sangat sesuai untuk aplikasi dengan pola permintaan tidak tetap, khususnya prototipe *AI* seperti *text-to-image generation*.

4.11. Pembahasan terhadap Performa Sistem

Hasil pengujian performa menunjukkan bahwa sistem mampu memberikan *latency* rata-rata 10.0–12.5 detik, yang masih wajar untuk aplikasi generatif berbasis komputasi tinggi. Waktu *cold start* sebesar 315–410 ms juga konsisten dengan karakteristik platform *serverless* modern sebagaimana dijelaskan dalam penelitian [6][20], sehingga dampaknya tidak signifikan pada penggunaan prototipe.

Pada *load test*, peningkatan *error rate* $\leq 0.8\%$ menunjukkan bahwa arsitektur *serverless* tetap mampu menangani permintaan simultan untuk skala kecil hingga menengah, mendukung temuan penelitian sebelumnya [4][5]. Secara keseluruhan, hasil ini menguatkan hipotesis bahwa kombinasi *serverless* dan *Gemini API* memberikan performa yang stabil serta terukur untuk implementasi sistem *text-to-image*.

4.12. Pembahasan terhadap Keamanan Sistem

Penggunaan *environment variables* sebagai tempat penyimpanan kredensial *API* terbukti efektif dalam mencegah kebocoran data. Cara ini mengikuti rekomendasi literatur keamanan *cloud-native* [15][7], yang menyatakan bahwa penyimpanan kredensial tidak boleh dilakukan pada sisi klien atau pada berkas statis.

Sanitasi *input* menambah lapisan keamanan untuk mengurangi risiko *injection*. Meskipun sistem ini tidak menyimpan data pengguna, praktik *input sanitization* telah direkomendasikan dalam berbagai publikasi mengenai keamanan aplikasi *web*. Dengan demikian, sistem dinilai aman dari risiko umum yang muncul pada aplikasi berbasis *API*.

4.13. Pembahasan terhadap Kualitas Gambar

Model *difusi* telah terbukti sebagai pendekatan paling efektif dalam *AI generatif* modern [1][13][14]. Hasil gambar yang dihasilkan menunjukkan kualitas detail yang baik dan sesuai *prompt*, membuktikan bahwa pustaka *@Google/genai* mampu memanggil *model generatif* dengan stabilitas tinggi.

Perbandingan dengan penelitian lain menunjukkan performa serupa dengan *model difusi* komersial seperti *Stable Diffusion* dan *Midjourney*, yang juga mengandalkan kemampuan generalisasi *difusi*. Kelebihan *Google Gemini* terletak pada integrasi *API* yang terstandardisasi, struktur respons yang mudah diproses, serta optimasi internal tanpa perlu pengaturan model sendiri, menguatkan kesesuaian *difusi* untuk aplikasi *text-to-image generation* di berbagai domain [11][13].

4.14. Kontribusi Penelitian terhadap Literatur Eksisting

Penelitian ini memberikan tiga kontribusi utama:

a. Demonstrasi integrasi *AI generatif* pada arsitektur *serverless*
   Belum banyak penelitian yang secara eksplisit membahas integrasi *serverless* dengan layanan *generatif* berbasis model *difusi*. Penelitian ini memperkuat literatur *serverless* [4][6][8], dengan memberikan studi kasus konkret.
b. Model implementasi minimalis namun aman
   Dengan penggunaan *@Google/genai* dan *environment variables*, penelitian ini menunjukkan bagaimana sistem dapat dibangun dengan aman tanpa infrastruktur kompleks. Ini mendukung teori keamanan *cloud* [15][7].

c. Pembuktian stabilitas performa *serverless* untuk permintaan AI generative
   Hasil menunjukkan bahwa *serverless* dapat mendukung beban ringan hingga menengah dengan performa yang stabil, sesuai dengan literatur namun dalam konteks yang lebih spesifik.

4.15. Keterbatasan Penelitian

Beberapa keterbatasan yang perlu dicatat:

a. Sistem tidak menyimpan riwayat *prompt* sehingga tidak memungkinkan evaluasi longitudinal.
b. Pengujian performa dilakukan dalam skala terbatas.
c. *Output* gambar harus dikonversi ke hitam-putih dalam publikasi meskipun model menghasilkan gambar berwarna.

4.16. Implikasi dan Potensi Pengembangan

Keberhasilan prototipe ini membuka peluang untuk:

a. Penambahan *prompt history logging*.
b. Integrasi basis data atau *cache*.
c. Pengembangan *dashboard* pengguna.
d. Integrasi teknik *prompt engineering*.
e. Optimasi performa dengan *edge functions*.

5. KESIMPULAN DAN SARAN

Penelitian ini berhasil merancang dan mengimplementasikan prototipe sistem *Text-to-Image* berbasis *Google Gemini* dengan arsitektur *serverless* menggunakan *Vercel Functions*, yang aman bagi kredensial, efisien dalam proses generatif, dan skalabel di lingkungan produksi. Sistem ini menunjukkan integrasi *frontend* web dan *backend serverless* yang aman, dengan respons stabil di bawah 10 detik. Pengujian fungsional dan performa membuktikan *pipeline generatif* berjalan *end-to-end*, dari validasi *prompt*, pemrosesan oleh *Gemini API*, hingga penyajian citra dalam format *Base64*. Meski demikian, penelitian ini tergantung pada layanan pihak ketiga dan belum dilakukan *stress test* untuk menilai skalabilitas tinggi. Penelitian selanjutnya bisa meningkatkan sistem melalui *caching*, *prompt optimization*, komparasi antar model *generatif* (misalnya *Stable Diffusion* atau *DALL-E*), serta pengujian pada berbagai perangkat dan jaringan. Secara keseluruhan, penelitian ini membuktikan arsitektur *serverless* efektif, aman, dan fleksibel untuk sistem *text-to-image* generatif berbasis *AI* di era *cloud computing* modern.

DAFTAR PUSTAKA

[1] C. Zhang, C. Zhang, M. Zhang, I. S. Kweon, and J. Kim, "Text-to-Image Diffusion Models in Generative AI: A Survey," *arXiv*, pp. 1–40, 2024.

[2] H. Shen *et al.*, "Efficient Diffusion Models : A Survey," *arXiv*, pp. 1–50, 2025.

[3] Z. Li, J. Li, L. Xiong, Z. Fu, and Z. Li, "A Comprehensive Survey on Visual Concept Mining in Text-to-image Diffusion Models," *arXiv*, 2025.

[4] S. Kounev *et al.*, "Serverless Computing: What It Is, and What It Is Not ?," *Commun. ACM*, vol. 66, no. 9, pp. 80–92, 2023, doi: 10.1145/3587249.

[5] S. M. Fati and M. Alenezi, "Transforming Application Development With Serverless Computing," *Int. J. Cloud Appl. Comput.*, vol. 14, no. 1, pp. 1–16, 2024, doi: 10.4018/IJCAC.365288.

[6] M. Akour and M. Alenezi, "Reducing Environmental Impact with Sustainable Serverless Computing," *MDPI*, vol. 17, no. 7, pp. 1–23, 2025, doi: 10.3390/su17072999.

[7] K. Ni, S. K. Mondal, H. M. D. Kabir, T. Tan, and H.-N. Dai, "Toward security quantification of serverless computing," *J. Cloud Comput. Adv. Syst. Appl.*, vol. 13, no. 140, pp. 1–27, 2024, doi: 10.1186/s13677-024-00703-y.

[8] P. Creswell, *Serverless Computing: Principles, Practices, and Patterns*, 1st Edition. California: O'Reilly Media, 2022.

[9] Sugiyono, *Metode Penelitian dan Pengembangan (Research and Development/R&D)*, Revisi 2021. Bandung: Alfabeta, 2021.

[10] A. Rashid and C. Seitz, *Building Serverless Applications with JavaScript: A Practical Guide*, 1st Edition. Birmingham: Packt Publishing, 2023.

[11] S. Fang, "A Comprehensive Survey of Text Encoders for Text-to-Image Diffusion Models," *EAI Endorsed Trans. AI Robot.*, vol. 3, pp. 1–11, 2024, doi: 10.4108/airo.5566.

[12] Y. Liang *et al.*, "Rich Human Feedback for Text-to-Image Generation," *arXiv*, pp. 19401–19411, 2024, doi: 10.48550/arXiv.2312.10240.

[13] M. Chen, S. Mei, J. Fan, and M. Wang, "Opportunities and challenges of diffusion models for generative AI," *Natl. Sci. Rev.*, vol. 11, pp. 1–23, 2024, doi: 10.1093/nsr/nwae348.

[14] H. Chen, Q. Xiang, J. Hu, M. Ye, C. Yu, and H. Cheng, "Comprehensive exploration of diffusion models in image generation: a survey," *Springer J. Intell. Fuzzy Syst.*, vol. 58, no. 99, pp. 1–49, 2025, doi: 10.1007/s10462-025-11110-3.

[15] M. Alshaikh, W. Alsemaih, S. Alamri, and Q. Ramadan, "Using Supervised Learning to Detect Command and Control Attacks in IoT," *Int. J. Cloud Appl. Comput.*, vol. 14, no. 1, pp. 1–16, 2024, doi: 10.4018/IJCAC.334214.

[16] V. K. Thatikonda, "Serverless Computing: Advantages, Limitations and Use Cases," *Eur. J. Theor. Appl. Sci.*, vol. 1, no. 5, pp. 341–347, 2023, doi: 10.59324/ejtas.2023.1(5).25.

[17] M. Ghorbian and M. Ghobaei-Arani, "Serverless Computing: Architecture, Concepts, and Applications," *arXiv*, 2025, doi: 10.48550/arXiv.2501.09831.

[18] M. Sanderson and D. Pieter Kroese, *Hands-On Generative AI with Diffusion Models: Theory and Practice*, 1st Edition. Birmingham: Packt Publishing, 2024.

[19] J. Wen, Z. Chen, X. Jin, and X. Liu, "Rise of the Planet of Serverless Computing: A Systematic Review," 2022.

[20] N. K. Prajapati, "Cloud-based serverless architectures: Trends, challenges and opportunities for modern applications," *World J. Adv. Eng. Technol. Sci.*, vol. 16, no. 01, pp. 427–435, 2025, doi: 10.30574/wjaets.2025.16.1.1225.